BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images,
Please do not report the images to the
Image Problem Mailbox.

Re: Some thoughts on streaming

Holger Grahn (*holger@blaxxun.com*)
Sun, 14 Jun 1998 20:48:34 +0100

- **Messages sorted by:** [date][thread][subject][author]
- **Next message:** David Chamberlin: "Re: Comments on streaming thoughts."
- **Previous message:** Rex Brooks: "Re: Comments on streaming thoughts."
- **Maybe in reply to:** Bernie Roehl: "Some thoughts on streaming"

Thank you all for bringing up the "Some thoughts of streaming"
thread,
here are some collected ideas :

To read as html http://www.snafu.de/~hg/stream.html

Ideal solution MPEG 4
==========
MPEG 4 already describes solutions for the following items

□binary encoding of a VRML scene graph
□dynamic Node and Node field update
□streaming of the scene graph
□streaming of audio, video and images
□interleaving / multiplexing the different streams into
one bit stream
□VRML like authoring language

Additional MPEG4 defines additional nodes for

□facial animation
□synthetic sound
□2D VRML
□Layouts / Composition

By adopting/defining a pure MPEG4 VRML 97 subset,
MPEG4 could be the binary / streaming file format for VRML.

MPEG 4 sample : dynamic scene graph update with nodes

```
DEF GRP Group {
 children [
     DEF TS TimeSensor { enabled FALSE }
      ...
  ]
}

AT 2000 {
 APPEND TO GRP.children
  Transform {
```

```
    ...
  }
}
```

AT 3000 REPLACE TS.enabled BY TRUE

Existing solutions or ideas

blaxxun community server/CCpro VRML shared events.

Basic VRML field types can be routed from a client VRML scene via the network to other clients. Server side automated characters (Bots), or server side scripts can stream events to clients, or act depending on events coming from the clients.

```
DEF SharedZone BlaxxunZone {
 events [
  DEF SharedRotation SharedEvent { name "newRotation" }
  DEF SharedTranslation SharedEvent { name "newTranslation" }
 ]
}

DEF T Transform {

}
ROUTE SharedRotation.rotation_changed TO T.set_rotation
ROUTE SharedTranslation.translation_changed TO T.set_translation
```

Sending to the network can be done by routing to the set_type eventIn of a SharedEvent.

Bernies idea

Bernies idea as described in http://ece.uwaterloo.ca/~broehl/streams/proposal.html

For some type of applications it could be useful but I agree with David Chamberlain: Decoding must be done using a Java, there is no authoring time support for defining streamed data, the MovieTexture node gets over bloated.

If encoding and decoding is propiertary to the application anyway , by adding a RawAudio node, video & audio data could be streamed using a PixelTexture and a RawAudio node from Java or the EAI.

VRML Stream node

Here is an idea for a simple, low-level VRML Stream interface

```
Stream {
  exposedField MFString url   # HTTP / RTSP / UDP  / TCP?
  exposedField SFString contentType  # the format of the data stream
  exposedField SFTime startTime # for starting the stream
```

```
  exposedField SFTime stopTime # for stopping the stream
  exposedField SFBool loop TRUE # looping
  eventOut SFBool isActive # notification for start & stop  of stream

  # delivery of stream data
  eventOut SFImage   image_changed # contentType image/* movie/*

  eventOut MFInt32  audioFormat_changed #   waveformat structure : channels, samples per second,
bits per sample, encoding
  eventOut MFInt32   audioData_changed  # raw audio PCM data packages

  # streaming a VRML with each top level node as unit
  eventOut SFNode  node_changed

  # streaming basic field types
  eventOut SFFloat float_changed # contentType = "vrml/SFloat"
  eventOut MFFloat mffloat_changed # contentType = "vrml/MFFloat"
  ......

}

Together with a RawAudio node similar to David's Idea:
AudioBuffer {
  exposedField  MFInt32   format #   waveformat structure : channels, samples per second, bits per
sample, encoding
  eventOut MFInt32   data  # raw audio data packages corresponding to encoding

  ... / startTime / stopTime/ loop / lowWater ??
  eventIn SFFloat addSilence
  eventIn SFTime flush

}
Usage scenario:

Reading a image from some data file :
DEF S Stream {
  url "myimage.jpg"
}
DEF T PixelTexture {}
ROUTE S.image_changed TO T.set_image

A Text ticker reading text strings from cgi-bin script

DEF S Stream {
  url "/cgi-bin/text.cgi"
  contentType "vrml/MFString"
}
DEF T Text {}
ROUTE S.string_changed TO T.set_string

Reading a movie
```

```
DEF S Stream {
  url "mymovie.mpg"
}
DEF T PixelTexture {}
ROUTE S.image_changed TO T.set_image

DEF A AudioBuffer {}
ROUTE S.audioFormat_changed TO A.set_format
ROUTE S.audioData_changed TO A.set_data
```

If the contentType is empty, the browser could derive it from the mime type of the URL.
Supported formats would be raw-audio / wav, still and movie image formats.
For contentType = vrml/a-vrml-field-type
The data stream would simply field values coded in VRML utf8 ascii encoding :
e.g.
contentType vrml/MFFloat
data :
0.5
[1.3 2.6]
0.7

for contentType vrml/*
the field values could be preceded with the eventOut name e.g.
float 0.5
rotation 0 0 1 3.4
node Material { diffuseColor 0.3 }
MFVec3f [1 2 3, 4 5 6]

by adding a keyword "AT timeOffset" a relative event delivery time offset could be set.
AT 0
float 0.5
AT 3 // 3 seconds later
float 0.7

Once a VRML binary encoding standard is available, the first line of the data file would describe the encoding
e.g. #VRML 2.0 MPEG4

This idea is sounding relatively cheap to implement, seems to be flexible for many usage scenario, especially in combination with cgi-bin programs or direct TCP connections.

Defining more complex streams with several interleaved data types, may need a sort of channel description and the channel information is encoded in the data stream
A very rough sketch :

```
Stream {
  url "rtsp: ..."
  channels [
      DEF Movie1 Channel {
```

```
            target USE MovieTexture1
        }
        DEF Audio1 Channel {
            target USE AudioClip1
        }
        DEF Audio2 Channel {
            target USE AudioClip2
        }
        DEF Motion1 Channel {
            target USE MotionPacket1
        }
    ]
}
PROTO MotionPacket {
    exposedField SFVec3f position
    exposeField SFRotation orientation
}
DEF MotionPacket1 MotionPacket {}
ROUTE MotionPacket1.position_changed TO myObject.set_translation
ROUTE MotionPacket1.orientation_changed TO myObject.set_rotation
```

The stream node, recognizes some native Browser nodes (MovieTexture, ImageTexture, AudioClip, Anchor) to automatically route the network data to the nodes internal streaming input pin.
For other nodes the data format is described in a Proto interface.

In MPEG4 there are object and entity stream descriptors for this purpose.

SMIL
====

Defining sequences of animation's is currently quite authoring intensive in VRML.
E.g. something like

```
play animation 1
at end of animation 1 start animation 2 with a 3 second delay
parallel play movie1 after that sound 1
```

Several Scripts / TimeSensors must be controlled to get the effect.
SMIL concepts could be expressed in VRML using new nodes :

```
DEF MyAnimation   Par {
    children [
    Seq {
        children [
          DEF animation1Timer TimeSensor { cycleInterval 20 }
          DEF waiter1 TimeSensor { cycleInterval 3 }
          DEF animation2Timer TimeSensor { }
        ]

    }
    Seq {
```

```
        children [
          DEF movie1 MovieTexture {}
          DEF sound1 AudioClip {}
        ]
    }
    ]
  }
ROUTE someSensor.start TO MyAnimation.set_startTime
```

Seq and Par are controlling time dependent nodes, which have an set_startTime SFTime eventIn, and a possibility to compute a duration, or monitoring an isActive FALSE eventOut.

TV or not TV ?

I think its important to not understand under "VRML streaming" only a data downstream for yet another TV like presentation.
Streaming should be possible in both directions, for multi-media communication or multi-user usage scenarios.

-- Holger

- **Next message:** David Chamberlin: "Re: Comments on streaming thoughts."
- **Previous message:** Rex Brooks: "Re: Comments on streaming thoughts."
- **Maybe in reply to:** Bernie Roehl: "Some thoughts on streaming"